Matti Ikäheimonen

ACCESS-SUUNNITTELUTIETOKANNAN JA AUTOCADIN VÄLINEN TIEDONSIIRTO

Opinnäytetyö
Sähkötekniikan koulutusohjelma

Toukokuu 2011

MIKKELIN AMMATTIKORKEAKOULU
Mikkeli University of Applied Sciences

KUVAILULEHTI

MIKKELIN AMMATTIKORKEAKOULU Mikkeli University of Applied Sciences	**Opinnäytetyön päivämäärä** 9.5.2011
Tekijä(t) Matti Ikäheimonen	**Koulutusohjelma ja suuntautuminen** Sähkötekniikan koulutusohjelma
Nimeke Access-suunnittelutietokannan ja AutoCADin välinen tiedonsiirto	

Tiivistelmä

Tämän opinnäytetyön tarkoituksena on kehittää sähköistyksen ja instrumentoinnin kenttäsuunnittelussa käytettävän tietokantaohjelman ja AutoCAD –ohjelman välistä linkitystä tiettyjen asiakasprojektien tehostamiseksi. Insta Automation Oy:lle tekemäni ohjelmistokehitystyön tavoitteena on saada oikeinkirjoitettu tieto siirtymään Excel -taulukoista valmiisiin piirikaavioihin mahdollisimman pienellä manuaalisella käsittelyllä kirjoitusvirheiden välttämiseksi. Tiedonsiirtoa AutoCAD –ohjelmaan on mahdollista hyödyntää myös muissa suunnitteluprojekteissa.

Asiakkaalta saaduista Excel –taulukoista muodostetaan suunnittelutietokanta, jota hallitaan Microsoft Access tietokantaohjelmalla. Accessin toimintoja on laajennettu oleellisesti VBA –koodilla, johon on upotettu SQL:ää kyselyiden helpottamiseksi. Tiedonsiirron perustana kehitystyössä käytetään tyyppikaavioita, joihin halutut tiedot korvataan. Tyyppikaavioita voi olla useita ja niitä voidaan lisätä projektin niin vaatiessa. Korvauksessa käytetään srxtext –ilmaistyökalua, jolla kuvasta etsitään siihen muokattu mallitagi ja korvataan se halutulla tiedolla. Tiedonsiirtäminen tietokannasta AutoCAD:iin tapahtuu kahdessa vaiheessa. Ensimmäinen vaihe päivittää käyttäjän valitseman piirin mukaiset tiedot tiedonsiirtotauluun. Toinen vaihe tekee tiedonsiirtotaulun perusteella scriptin, jolla tiedot saadaan siirrettyä AutoCAD:iin.

Asiasanat (avainsanat)

AutoCAD, Access, SQL, VBA

Sivumäärä	**Kieli**	**URN**
19+ liitteet 8	Suomi	

Huomautus (huomautukset liitteistä)

Ohjaavan opettajan nimi Hannu Honkanen	**Opinnäytetyön toimeksiantaja** Insta Automation Oy / Timo Kolari

DESCRIPTION

MIKKELIN AMMATTIKORKEAKOULU Mikkeli University of Applied Sciences	**Date of the bachelor's thesis** 9 May 2011
Author(s) Matti Ikäheimonen	**Degree programme and option** Electrical Engineering

Name of the bachelor's thesis

Data transfer between Access designing database and AutoCAD

Abstract

The aim of this Bachelor's Thesis is to develop a link between database and AutoCAD software used in electrical and instrumentation designing. This development project is done for making designing to be more effective for specific client projects. Objective of this development project is to transfer correctly written information from the Excel spreadsheets to the finished circuit diagrams. The transfer should be done with a minimum of manual handling to avoid typing errors. Data transferring to the AutoCAD is also available for other designing projects.

The designing database managed by Microsoft Access database program is created from Excel spreadsheets received from customer. Access program's functions have been significantly extended by VBA code, which is embedded with SQL programming language to facilitate queries. Data transmission is based on the type charts where the data will be replaced. Srxtext-freeware tool is used for replacing. With srxtext-tool model text tags are sought and they are replaced with the wanted information. Data transfer from the database to the AutoCAD is done in two steps. The first step updates the circuit information chosen by user to the data transfer table. In the second step data transfer table information is converted to a script, which makes data transfer to the AutoCAD possible.

Subject headings, (keywords)

AutoCAD, Access, SQL, VBA

Pages	**Language**	**URN**
19+ appendices 8	Finnish	

Remarks, notes on appendices

Tutor	**Bachelor's thesis assigned by**
Hannu Honkanen	Insta Automation Oy / Timo Kolari

SISÄLTÖ

1 JOHDANTO

Tämä opinnäytetyö on saanut alkunsa ohjelmistokehitystarpeesta, jonka toteuttaminen ei ole mahdollista normaalin työn ohessa. Ohjelmakehityksen tarkoituksena on kehittää sähköistyksen ja instrumentoinnin kenttäsuunnittelussa käytettävän tietokanta- ja AutoCAD –ohjelman välistä linkitystä, käytännön toteuttamista sekä käytettävyyden parantamista tiettyjen asiakastoimitusten suunnittelutyön tehostamiseksi. Kehitystyön tavoitteena on saada oikeinkirjoitettu tieto siirtymään valmiisiin piirikaavioihin Excel –taulukoista, mahdollisimman pienellä manuaalisella käsittelyllä kirjoitusvirheiden välttämiseksi.

Raportissa kerron kuinka tietoa saadaan siirrettyä AutoCAD –ohjelmaan ja kuinka tiedonsiirto on toteutettu tässä kehitystyössä. Kerron myös ohjelmistokehitystyössä käytetyistä tietokantaohjelmoinnin menetelmistä, jotka näen tarpeelliseksi vastaavien projektien toteuttamisessa.

Opinnäytetyöni toimeksiantajana on Insta Automation Oy. Yritys on erikoistunut teollisuuden ja prosessien sähköautomaation suunnittelu-, valmistus-, asennus- ja ylläpitopalveluihin sekä kokonaistoimituksina että erillisinä palveluina. Kehitystyö on ollut tarkoitus tehdä normaalintyön ohessa, mutta se vaatii aikaa ja paneutumista, eivätkä yrityksen omat työntekijät ole voineet sitä toteuttaa. Tilanne tarjosikin minulle hyvän mahdollisuuden tutustua sähköistyksen ja instrumentoinnin kenttäsuunnittelutyöhön ja siinä käytettäviin menetelmiin.

Teen työtäni Varkauden toimistolle, joka kuuluu Insta Automation Oy:n aluetoimintayksikköön ja tuottaa automaation sekä sähköistyksen konsultointi- ja suunnittelupalveluja, asennusvalvontaa ja testauspalveluja sekä kokonaistoimituksia. Suunnitteluosaston aluepäällikkönä toimii Varkaudessa Timo Kolari, jota haluan erityisesti kiittää opinnäytetyöni mahdollistamisesta.

2 TOTEUTUS JA RATKAISUT

Työn toteuttaminen alkoi tavallaan loppupäästä, eli tiedonsiirto mahdollisuuksien kartoittamisella AutoCAD:iin. Tämä tuntui järkevältä, koska tiedonsiirtoon käytettävien tietojen keräämistä ja käsittelyä voisi miettiä myöhemmin. Tiedonsiirron periaatteena AutoCAD:iin toimivat tyyppikaaviot. Tyyppikaavio näyttää muuten valmiilta kaaviolta, mutta sen tekstitagit on muutettu vastaamaan tiedonsiirtotaulun vastaavia tageja. Esimerkkinä tyyppikaavio T1(liite 1), jota käytetään lämpötilanmittauspiirien kuvaamiseen.

Tiedonsiirto AutoCAD:iin tulisi siis toimimaan niin, että tyyppikaaviosta etsitään siihen muokattu mallitagi ja se korvataan halutulla tiedolla. Ratkaisuksi tämän etsi/korvaa –toiminnon suorittamiseen löytyi ilmaistyökalu Srxtext, josta on kerrottu tarkemmin seuraavan otsikon alla. Työkalua käyttääkseen piti kokeilemalla selvittää, miten etsi ja korvaa –toiminto AutoCAD:ssä toimii, ja kuinka sille saadaan annettua oikeat määritykset. Kokeileminen alkoi manuaalisesti kirjoitetuilla scripteillä ja hyvin pelkistetyllä kaaviolla, jossa oli vain joitakin tekstitageja.

Kun scriptien vaatima rakenne selvisi kokeilemalla, piti keksiä ratkaisu, jolla tiedonsiirtotaulun tiedoista saatiin kirjoitettua scripti. Scriptien muodostamisesta ja toiminnasta on kerrottu tarkemmin otsikoiden 3.3 ja 3.4 alla.

2.1 Srxtext –työkalu

Srxtext on ilmaistyökalu, jonka avulla voidaan etsiä ja korvata AutoCAD -kuviin kirjoitettuja TEXT-, MTEXT-, DIMTEXT-, ATTRIB- ja ATTDEF –tyyppisiä tekstejä. Työkalu on mahdollista ladata osoitteesta http://www.xanadu.cz ja se toimii AutoCAD:n versioissa 2000-2010.

2.2 Tietojen kerääminen

Kun tiedonsiirto AutoCAD:in ja tiedonsiirtotaulun välillä toimi halutulla tavalla piti alkaa miettimään, kuinka tieto saataisiin siirrettyä tiedonsiirtotauluun. Lähdetietoja on viidessä eri taulussa, joten ensimmäisenä tuli mieleen niiden yhdistäminen yhdeksi tauluksi. Tämä kuitenkin osoittautui hankalaksi, koska tauluissa ei ole selkeätä

yhteistä tekijää, jolla tietojen linkittäminen onnistuisi. Luovuin ajatuksesta ja aloin miettiä kuinka tieto saataisiin siirrettyä suoraan tiedonsiirtotauluun, jokaisesta taulusta erikseen. Lähdetietojen keräämisen suunnittelussa minulla oli apuna viimeisimmän asiakasprojektin tiedot, joiden avulla sain kokonaiskuvan kerättävistä tiedoista. Lähdetietojen käsittelystä tarkemmin seuraavan otsikon alla.

Tietojen kerääminen onnistuu helposti SQL kyselyllä. Kysely voidaan toteuttaa jokaiselle taululle erikseen, käyttäjän valitseman piiritunnuksen mukaan. Lisäksi on huomioitava liittimien numerointi logiikkakortissa, joka on toteutettu kahdella erillisellä taululla. Toisessa taulussa on liitinnumerointi passiiviselle ja toisessa aktiiviselle virransyötölle. Muiden kyselyiden jälkeen kortti- ja virransyöttötyypin sekä kanavan mukaan tehdään toinen kysely, jolla haetaan liitinnumerot. SQL:ää hyödynnetään myös kyselyiden tulosten päivittämiseen tiedonsiirtotauluun. SQL –periaatteista enemmän otsikon 4.3 alla.

Tähän asti kokeilua oli tehty vain yhdellä tyyppikaaviolla ja sitä vastaavalla tiedonsiirtotaululla. Nyt mukaan tuli useampi tyyppikaavio, joka tarkoitti myös useampaa tiedonsiirtotaulua. Tyyppikaavioissa on paljon vastaavuuksia, mutta myös eroavaisuuksia, minkä vuoksi jokaiselle tiedonsiirtotaululle oli tehtävä oma kysely/päivitys -proseduuri. Vastaavuuksien ansiosta tässä voidaan hyödyntää paljon edellisissä proseduureissa käytettyä koodia, joka nopeuttaa uusien kysely/päivitys –proseduurien tekemistä. Tyyppikaavio, tiedonsiirtotaulu ja kysely/päivitys –proseduuri on jokainen nimetty samalla tavalla, mikä mahdollistaa yhden tiedon käyttämisen monessa paikassa. Tämä myös helpottaa ohjelmointia ja vähentää virheen mahdollisuutta.

2.3 Lähdetietojen käsittely

Lähdetiedot saadaan asiakkaalta Excel –taulukkoina, jotka pitää siirtää suunnittelutietokantaan. Tässä on käytetty yhtä Excel –taulukkoa, jossa jokaiselle lähdetietotaulukolle on oma välilehtensä. Jokaisella välilehdellä on omat kiinteät otsikkorivit, jotta tietojen linkittäminen onnistuu tietokannassa. Otsikointi on tehty siten, että se vastaa alkuperäisiä taulukoita, joten tietojen siirtäminen onnistuu lähes suoraan kopioimalla.

Tämä toteutustapa jättää taulukoihin paljon ylimääräisiä sarakkeita, joista ei tarvita tietoja. Kuitenkin rivimäärät ovat niin pieniä, ettei ylimääräisillä tiedoilla ole toimivuuden kannalta mitään merkitystä. Taulukoihin pitää vielä lisätä käytettävät tyyppikaaviot ja piirustusnumerot, joten käsityötä jää riittävästi ilman ylimääräistä muokkaamistakin.

Tiedonsiirtoon käytettävä Excel –tiedosto on nimeltään LISTS.xls, joka sijoitetaan projektikansioon. Kun asiakkaalta saadut tiedot on siirretty ja tarvittavat muokkaukset suoritettu, voidaan tiedosto tuoda suunnittelutietokantaan. Tuonti suoritetaan käyttöliittymässä olevalla ”Tuo excel -taulut” –painikkeella. Samaa painiketta voidaan käyttää myös aiemmin tuotujen taulujen päivittämiseen tietokannassa.

2.4 Virheiden käsittely

Ihannetapauksessa Visual Basic –toimintosarjat eivät tarvitsisi lainkaan virheenkäsittelykoodia. Todellisuus kuitenkin sanelee, että laiteongelmat tai käyttäjän ennalta arvaamattomat toimenpiteet voivat aiheuttaa suorituksenaikaisia virheitä, jotka pysäyttävät ohjelman toiminnan, eikä käyttäjä tavallisesti pysty jatkamaan työskentelyä. Toiset virheet eivät ehkä pysäytä toimintaa, mutta ne saavat sovelluksen toimimaan ei-toivotulla tavalla. (Suominen 1997 ,344).

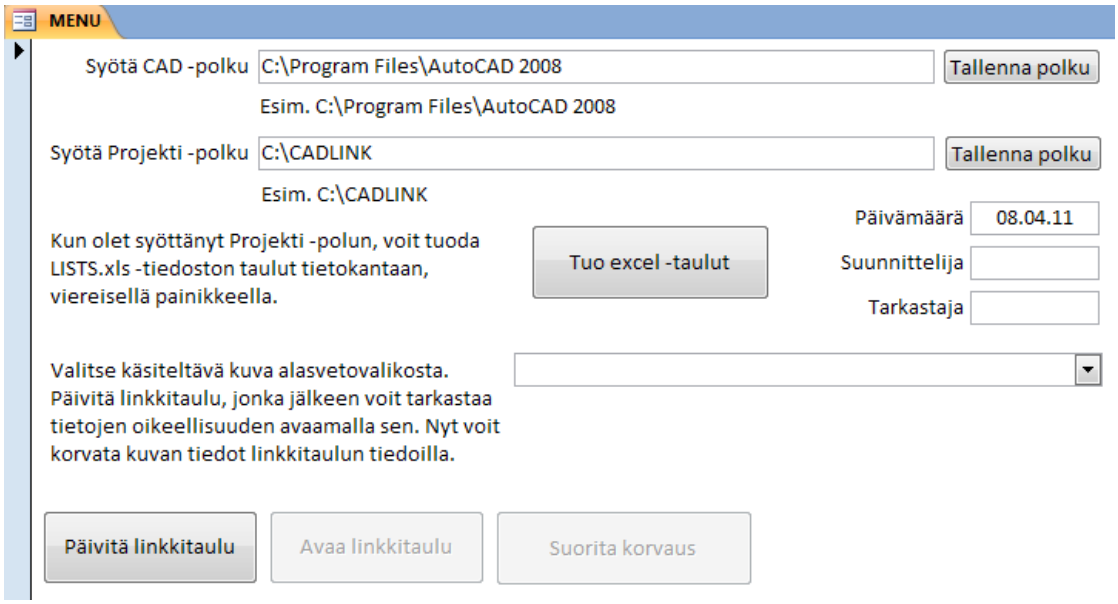
Virheiden käsittelyä lähdin toteuttamaan sen pohjalta, että käyttäjä ei saisi ohjelmaa pysähtymään. Mahdollisista puutteellisista syötteistä ja muista kokonaisuuden toimintaan liittyvistä virheistä ilmoitetaan virheilmoituksella. Virhetapauksissa toiminnot pysäytetään siten, että virheilmoituksen kuittaamisen jälkeen käyttäjä voi korjata virheen ja jatkaa työskentelyä normaalisti. Projekti- ja CAD- polkujen oikeellisuus tarkastetaan niiden syöttämisvaiheessa. Kyselyiden epäonnistumisesta ei käyttäjää informoida virheilmoituksella vaan tulosmuuttujien arvoksi asetetaan ”##Not found##” tai ”##Field is empty##”. Näin käyttäjä voi myöhemmin reagoida virheisiin.

2.5 Käyttöliittymä

Käyttöliittymän eli lomakkeen suunnittelun lähtökohtana oli saada siitä mahdollisimman yksinkertainen ja käytettävä. Käyttöliittymän tehtävänä on hallita

ohjelmakoodiin kirjoitettuja proseduureja ja suorittaa niitä käyttäjän valintojen mukaisesti. Lähtötilanteena käyttöliittymän toiminnan kuvaamisessa pidetään sitä, että LISTS.xls –tiedosto on saatu valmiiksi ja sijoitettu projekti –kansioon. Tämän tiedoston luonti onkin ainoa asia mitä käyttöliittymän avulla ei voida suorittaa.

Avattaessa käyttöliittymä, tekee se jo ensimmäiset toimintonsa. Se hakee viimeksi käytetyt polut PATHS –taulusta ja asettaa ne niille varattuihin tekstikenttiin. Myös ”Päivämäärä” –kenttään päivitetään ajantasainen tieto. Kuvassa 1 on kuvakaappaus käyttöliittymästä, sen avaamisen jälkeen.



KUVA 1. Käyttöliittymän kuvakaappaus

Mikäli polut eivät ole samat kuin edellisellä käyttökerralla täytyy ne syöttää uudelleen ja tallentaa tekstikentän vieressä olevalla painikkeella, kumpikin polku erikseen. Tallennuspainiketta painettaessa tarkastetaan polun oikeellisuus. CAD –polku hyväksytään mikäli kansiosta löytyy ”acad.exe” –tiedosto ja Projekti –polku hyväksytään mikäli kansio sisältää LISTS.xls –tiedoston. Polut tallennetaan jo aiemmin mainittuun PATHS –tauluun, josta niitä hyödynnetään myöhemmin.

Polkujen syötön jälkeen tuodaan lähdetietotaulut tietokantaan. Tämä tapahtuu ”Tuo excel –taulut” –painikkeella, jonka toiminta on aiemmin kerrottu otsikon 2.3 alla. Ilman lähdetietotauluja ei käyttöliittymällä pysty tekemään oikeastaan mitään, koska kaikki toiminnot perustuvat ”MEASURING_POINT_LIST –tauluun.

Alasvetovalikon tiedot tulevat MEASURING_POINT_LIST –tauluun tehtävästä kyselystä, joka näyttää piiritunnuksen ja sitä vastaavan kuvauksen muodostaman listan. Alasvetovalikon valinnan jälkeen voidaan päivittää tiedonsiirtotaulu ”Päivitä linkkitaulu” –painikkeella, kun ”Suunnittelija”- ja ”Tarkastaja” –tiedot on syötetty.

Kun tiedonsiirtotaulu on päivitetty, vapautuvat ”Avaa linkkitaulu”- ja ”Suorita korvaus” –painikkeet. Samat painikkeet taas lukkiutuvat mikäli alasvetovalikon valintaa muutetaan. Tämä tapahtuu sen vuoksi, että alasvetovalikon arvoa (value) käytetään monessa muussakin paikassa, eikä se saa muuttua tiedonsiirtotaulun päivittämisen jälkeen. ”Avaa linkkitaulu” –painikkeella voidaan tarkastaa päivitetty tiedonsiirtotaulu, että kaikki tarvittavat tiedot on päivittyneet. ”Suorita korvaus” –painikkeen toiminnasta on kerrottu tarkemmin otsikoiden 3.3 ja 3.4 alla.

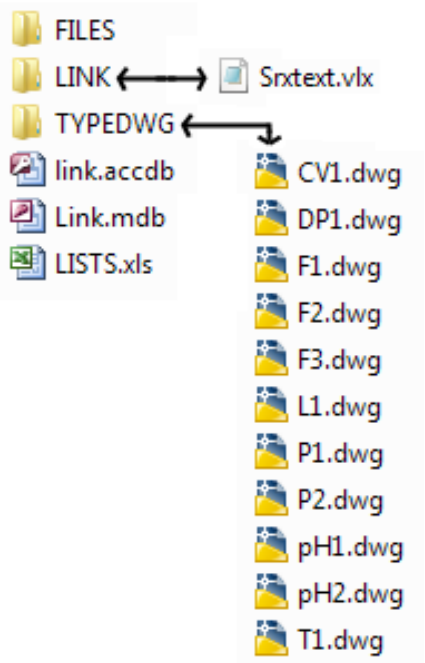
Kaikki käyttöliittymän painikkeet suorittavat proseduurin, jossa määritetään mitä painike tekee. Nämä proseduurit sijaitsevat Form_MENU –moduulissa, jota kutsutaan luokkamoduuliksi. Moduuleista kerrotaan tarkemmin otsikon 4.2.1 alla. Painikkeen proseduurilla voidaan kutsua yhteiskäyttömoduuleissa sijaitsevia proseduureja ja saman voi tehdä myös toisin päin.

3 TOIMINTA

Työssä käytettyyn Microsoft Access –tietokantaohjelmaan luodulla käyttöliittymällä voidaan hallita lähes kaikkia tarvittavia toimintoja, joita tiedonsiirtäminen edellyttää. Käyttöliittymästä kerrottiin aiemmin otsikon 2.5 alla. Seuraavat otsikot selventävät tarkemmin, mitä tiedon siirtäminen valmiisiin piirikaavioihin edellyttää.

3.1 Kansiorakenne ja tiedostot

Käyttöliittymän ja sen avulla tapahtuvan toiminnan perustana on oikea kansiorakenne sekä tarvittavat tiedostot. Projektissa käytettävän kansion nimellä ei ole merkitystä, mutta sen tulee sisältää kuvassa 2 olevat kansiot sekä tiedostot. Projektikansion sijainnillakaan ei ole merkitystä, koska sen polku voidaan määrittää käyttöliittymän avulla.

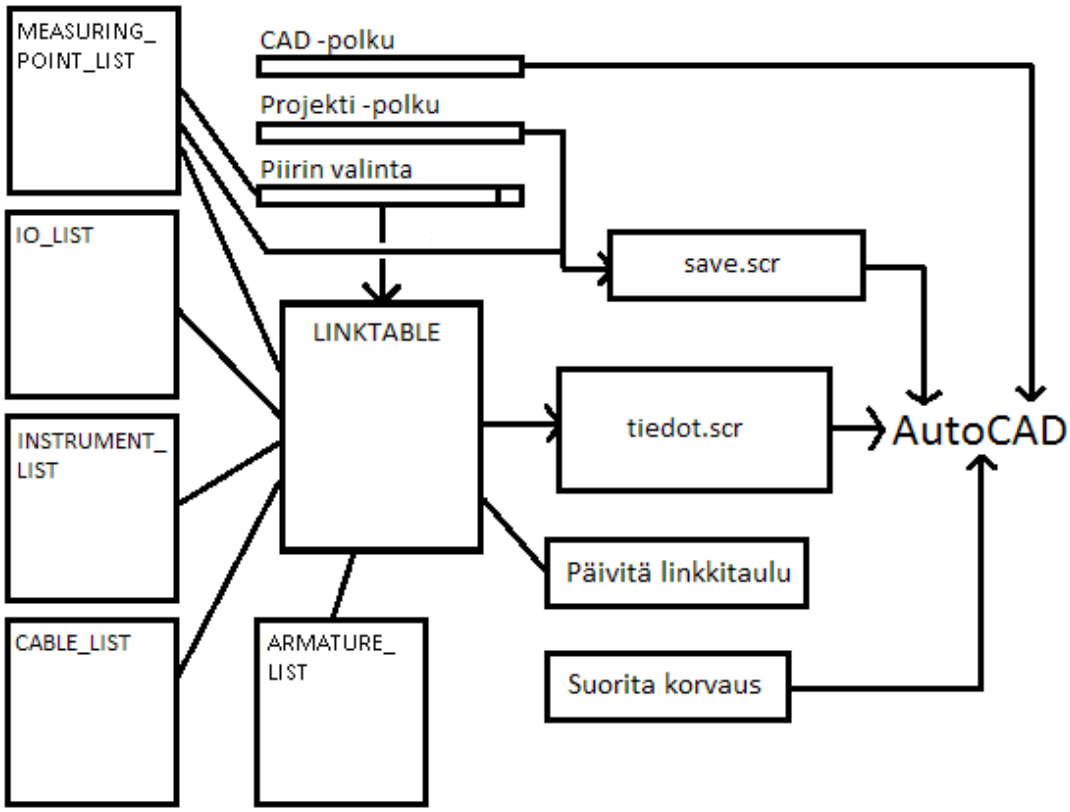


KUVA 2. Kansiorakenne sekä tarvittavat tiedostot

FILES –kansioon tallennetaan valmiit kaaviot. LINK –kansio on tiedonsiirrossa käytettäviä scriptejä varten ja se sisältää myös ”Srxtext.vlx” –työkalun. Scripteistä tarkemmin otsikon 3.3 alla. TYPEDWG –kansio sisältää kaikki tyyppikaaviot, joiden pohjalta valmiit kaaviot luodaan. Asiakkaalta saadut lähdetiedot on siirretty LISTS.xls –tiedostoon. ”Link.mdb” on varsinainen suunnittelutietokanta ja ”link.accdb” käsittää muut tietokantaobjektit.

3.2 Tiedonsiirto

Tiedonsiirto alkaa asiakkaalta saatujen tietojen käsittelyllä. Kuvassa 3 on esitetty periaatekuva tiedonsiirtymisestä lähdetietotauluista AutoCAD:iin. Lähtötilanteena tässä pidetään sitä, että lähdetietotaulut on jo tuotu suunnittelutietokantaan, kuten otsikon 2.3 alla on mainittu. Lähdetietotauluja ovat MEASURING_POINT_LIST, IO_LIST, INSTRUMENT_LIST, CABLE_LIST ja ARMATURE_LIST.



KUVA 3. Toiminnan periaatekuva

Tiedonsiirtäminen aloitetaan valitsemalla ”Piirin valinta” –alasvetovalikosta käsiteltävä piiri. Alasvetovalikko pitää sisällään ”MEASURING_POINT_LIST” –taulusta tehdyn kyselyn, joka näyttää piiritunnuksen ja sitä vastaavan kuvauksen. Valinnan jälkeen voidaan päivittää tiedonsiirtotaulu.

3.2.1 Tiedonsiirtotaulu

Tiedonsiirtotaulu, joka on kuvassa 3 nimellä ”LINKTABLE”, toimii ns. keräystauluna. Tämä taulu vaihtuu ja määräytyy sen perusteella mikä on ”Piirin valinta” -tietoa vastaava tyyppikaavio MEASURING_POINT_LIST –taulussa.

Jokaisella tyyppikuvalla on siis oma tiedonsiirtotaulunsa, joka määräytyy valitun piirin mukaan. Kun tiedonsiirtotaulu on määritetty, voidaan siihen käyttää koodiin kirjoitettua kysely/päivitys –proseduuria, joka toimii käyttöliittymässä olevalla ”Päivitä linkkitaulu” -painikkeella. Tämän proseduurin toimintaperiaatteesta tarkemmin otsikon 4.3.2 alla. Kysely/päivitys -proseduurilla haetaan tiedot lähdetietotauluista piiritunnuksen eli TAG Number:in perusteella, jotka sitten päivitetään tiedonsiirtotauluun niille määrättyyn sarakkeeseen. Tämä sarake näkyy kuvassa 4 nimellä ”Replace_with”.

T1

Replace_with	Find	Description
	c_address	Address
	equ_tag	Cable_from
	f_cable_position	Cable_position
	c_cabinet_tag	Cable_to
	f_cable_type	Cable_type
	c_comb_address	Combined_address
	p_date	Date
	p_designer	Designer
	p_drawing_number	Drawing_number
	p_inspector	Inspector
	i_manuf	Manufacturer
	i_meas_area	Measure_area
	c_bus_address	Profibus_address
	c_segment	Profibus_segment
	p_tag_name	TAG_Name
	p_tag_number	TAG_Number
	c_1	Terminal_I
	c_2	Terminal_II
	i_type	Type

KUVA 4. Tiedonsiirtotaulu tyyppikuvalle T1

Tiedonsiirtotaulu sisältää kaksi muuta saraketta, joiden tiedot eivät muutu missään vaiheessa. Sarake ”Find” sisältää tyyppikaaviota T1(liite 1) vastaavat tekstitagit, jotka korvausta tehtäessä etsitään kuvasta ja korvataan ”Replace_with” -sarakkeen tiedoilla. Viimeinen sarake ”Description” on korvattavien tietojen linkittämistä varten. Tämän sarakkeen avulla kyselyiden tiedot saadaan päivitettyä oikeille riveille.

3.3 Scriptien muodostaminen

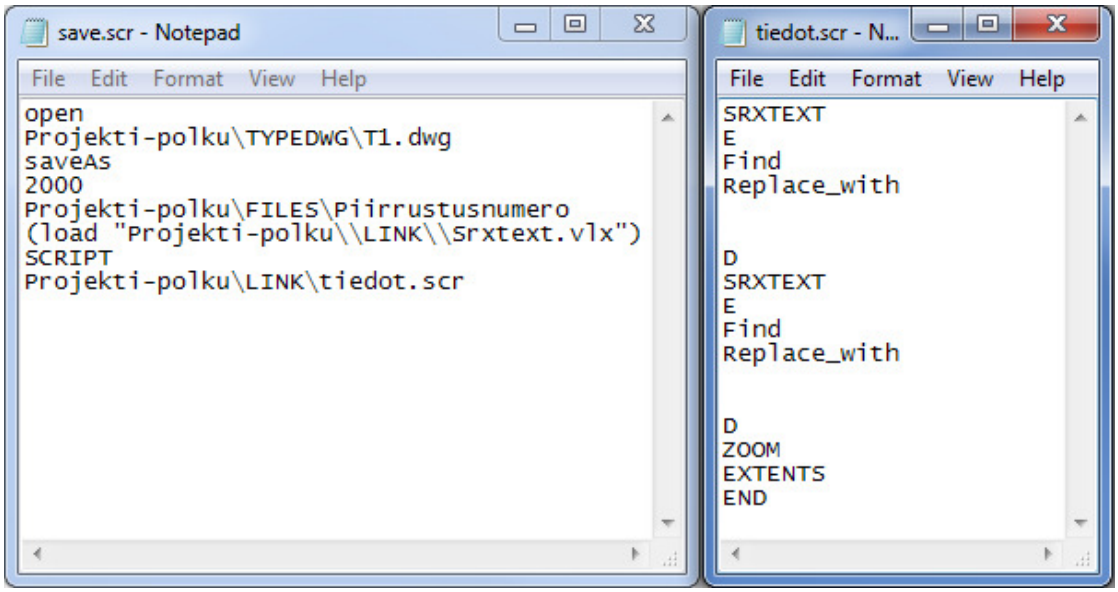
Tiedon siirtäminen AutoCAD:iin vaatii tiedonsiirtotaulun tietojen muokkaamista. Tämä tarkoittaa scriptien tekemistä, joilla AutoCAD:iin saadaan vietyä tieto etsittävistä ja korvaavista tiedoista. Etsi ja korvaa -toiminto saadaan Srxtext –työkalun

avulla, josta on kerrottu aiemmin otsikon 2.1 alla. Käyttäjän näkökulmasta tiedonsiirto tapahtuu yksinkertaisesti käyttöliittymän ”Suorita korvaus” –painikkeella. Tämä painike suorittaa ”Drive” –aliohjelman, joka suorittaa muut CADLINK –moduulin(liite 2) proseduurit oikeassa järjestyksessä. Seuraavassa on kerrottu vaiheittain, mitä liitteen 2 proseduureissa tapahtuu:

1. Suoritetaan ”GetLinkPath” –funktio, joka määrittää LINK –kansiolle polun. Polku asetetaan ”linkPath” –nimiseen muuttujaan, jotta sitä voidaan käyttää muissa toiminnoissa.
2. Suoritetaan ”DeleteTxtFiles” –aliohjelma, joka poistaa LINK –kansiosta mahdolliset vanhat save.txt ja tiedot.txt –tiedostot.
3. Suoritetaan ”MakeSaveTxt” –funktio, joka tekee save.txt –tiedoston. Funktio hakee käyttöliittymän alasvetovalikon valinnan arvon, josta se erottaa pelkän piiritunnuksen eli ”TAGNumber”:in. Tämän perusteella haetaan käytettävä ”linkTable” eli tiedonsiirtotaulu, ”GetLinkTable” –funktiolla. Piiritunnuksen perusteella suoritetaan myös kysely ”SQLQuery”, jolla haetaan piirustusnumero, joka toimii myös tallennusnimenä valmiille kuvalle. Suoritetaan ”GetProjectPath” –funktio, joka määrittää projekti –kansiolle polun. Projekti –polun perusteella voidaan määrittää muut tarvittavat polut, jotka tarvitaan save.txt –tiedoston kirjoittamiseen. Tässä funktiossa poistetaan myös mahdollinen vanha kuva, jotta scripti ei pysähdy kysymään olemassa olevan kuvan korvaamista. Lopuksi annetaan käsky suorittaa ”MakeInformationTxt” – aliohjelma, jolle annetaan määritykseksi ”linkTable”.
4. ”MakeInformationTxt” – aliohjelma avaa ”linkTable” –määrityksen mukaisen tiedonsiirtotaulun ja tekee siitä tietuejoukon ”recordSet”. Haetaan LINK –kansiolle polku, jonka jälkeen kirjoitetaan tiedot.txt -tiedosto. Tässä hyödynnetään ”Do until...Loop” –silmukkaa, jolla saadaan tietuejoukosta kaikki rivit kirjoitettua.
5. Suoritetaan ”DeleteScrFiles” – aliohjelma, joka poistaa LINK –kansiosta vanhat save.scr ja tiedot.scr –tiedostot.
6. Suoritetaan ”ChangeFilesToScr” – aliohjelma, joka muuttaa save.txt ja tiedot.txt –tiedostojen päätteet .scr:ksi.
7. Suoritetaan ”OpenCadAndSave” – aliohjelma, joka määrittää AutoCAD –ohjelmalle polun. Käynnistää AutoCAD:in ja aloittaa korvaamisprosessin suorittamalla save.scr –scriptin.

3.4 Etsi ja korvaa

Käynnistyttyään AutoCAD suorittaa save –scriptin. Tämä scripti avaa TYPEDWG –kansiosta valitun tyyppikaavion, jonka se tallentaa FILES –kansioon sille määritetyllä piirustusnumerolla. Tallennusmuotona käytetään ”AutoCAD 2000/LT2000 Drawing” –muotoa, yhteensopivuuden parantamiseksi. Ladataan Srxtext.vlx –työkalu, jolla etsi ja korvaa –toiminto saadaan käyttöön ja suoritetaan tiedot –scripti. Kuvassa 5 on save- ja tiedot –scriptien mallit, mitkä vastaavat työssä käytettyjä scriptejä. Piirustusnumero, joka on samalla tallennusnimi, on poistettu asiakastietojen turvaamiseksi.



KUVA 5. Save- ja tiedot –scriptien mallit

Tiedot –scripti sisältää kaiken varsinaisen tiedon, joka halutaan siirtää tyyppikaavioon. Srxtext –työkalu vaatii toimiakseen joitakin määrityksiä, jotka on lisätty varsinaisten tietuerivien lisäksi. Kun korvaukset on suoritettu, kuva keskitetään ”ZOOM” ja ”EXTENTS” –komennoilla. Kuvassa 5 tietuerivit on korvattu ”Find”- ja ”Replace_with” –teksteillä asiakastietojen turvaamiseksi. Tietuerivien määrä voi vaihdella tyyppikaaviosta riippuen, mutta scriptin rakenne pysyy aina samanlaisena.

4 OHJELMOINTI

Tämä otsikko kertoo työssä käytetyistä ohjelmista ja menetelmistä sekä niiden perusteista. Joitakin keskeisiä menetelmiä käsitellään tarkemmin, kuten tiedonsiirtotaulun päivitykseen käytettävää kysely/päivitys –proseduuria (4.3.2).

4.1 Microsoft Access –tietokanta

Relaatiotietokantaohjelmana tässä työssä on käytetty Microsoft Accessia. Tähän käyttötarkoitukseen olisi paljon muitakin vaihtoehtoja, mutta Microsoft Office –paketin mukana tuleva Access on nykypäivänä lähes jokaisessa työasemassa. Tämän vuoksi muita tietokantaohjelma vaihtoehtoja ei tarvinnut harkita.

Relaatiotietokanta (Relation Database) on järjestetty kokoelma toisiinsa liittyviä tietoja, joita on helppo käyttää tehokkaasti. Relaatiotietokannan käyttämiseen tarvitaan erityistä ohjelmaa, relaatiotietokantaohjelmaa (Relation Database Program), jota usein kutsutaan myös relaatiotietokannan hallintajärjestelmäksi (Relation Database Management System). (Sainio 2002, 6.)

Access-tietokanta sisältää tietojen lisäksi myös muita osia, joiden avulla voidaan suunnitella ja toteuttaa sovellus. Näin Access on relaatiotietokantaohjelman lisäksi myös sovelluskehitin, jolla voidaan toteuttaa yksinkertaisia sovelluksia jopa ilman ohjelmointia. (Sainio 2002, 8.)

Tässä työssä Microsoft Accessin ominaisuuksia pelkän tietokantaohjelman lisäksi ei paljonkaan ole hyödynnetty. Ainoastaan lomaketta on hyödynnetty, joka toimii ohjelman käyttöliittymänä, koska sitä ei ole järkevää toteuttaa pelkällä VBA –koodilla. Valitsin koodipainotteisen ohjelmointitavan siksi, että se on mielestäni helpompi toteuttaa ja mahdollistaa kommentoinnin. Kommentointi on tärkeää varsinkin tässä työssä, koska ohjelman jatkokehitys on todennäköistä, eikä tekijä välttämättä ole sama. Muutenkin kommentointi on tärkeää koodin ymmärtämisen kannalta, koska Visual Basic antaa käyttäjälleen monia eri mahdollisuuksia saman toiminnon kirjoittamiseen.

4.2 VBA

VBA eli Visual Basic for Applications, on siis Visual Basic –pohjainen ohjelmointikieli, joka toimii isäntäohjelman sisällä. Sainion (2002, 135) mukaan VBA on Office-perheen eräs keskeinen komponentti: sama ohjelmointikieli löytyy jokaisesta Office-tuotteesta. VBA on täysimittainen oliosuuntautunut ohjelmointikieli. Sillä voidaan laatia Accessin peruspalveluja hyvinkin monipuolisesti täydentäviä lisäratkaisuja. VBA:lla ohjelmointi tapahtuu Microsoft Visual Basic –editorilla, minkä vuoksi voidaankin puhua oikeastaan Visual Basic –ohjelmoinnista.

Kun Microsoft julkaisi vuonna 1991 Visual Basicin ensimmäisen version, se oli merkittävä helpotus vaikeana pidettyyn Windows-ohjelmointiin: aloittelijakin pystyi sen avulla luomaan toimivia ja tyylikkäitä ohjelmia. Uusien versioiden mukana Visual Basiciin on tullut monia parannuksia, ja se soveltuu laajojenkin ohjelmien tekemiseen. Enää Basic ei ole pelkästään aloittelijoille tarkoitettu harjoitteluohjelmointikieli, josta täytyy siirtyä pois taitojen kehittyessä. (Laaksonen 2002).

Visual Basicillä tehdään tapahtumaohjattuja ohjelmia. Tämä ratkottaa esimerkiksi painikkeen painalluksesta suoritettavaa toimintosarjaa eli proseduuria. Visual Basic on proseduuraalinenkieli, mutta siihen voidaan myös upottaa ei-proseduraalista SQL:ää. SQL –periaatteista kerrotaan tarkemmin otsikon 4.3 alla.

Sub-toimintosarja tarkoittaa komentojen Sub ja End Sub väliin sijoitettuja koodirivejä, joka suorittaa tietyn tehtävän, mutta ei palauta arvoa. Function-toimintosarja on koodikokonaisuus, joka on sijoitettu komentojen Function ja End Function väliin. Kuten Sub-toimintosarja, sekin suorittaa tietyn tehtävän, mutta toisin kuin ensin mainittu, se myös palauttaa arvon. (Suominen 1997 ,8).

Visual Basic –ohjelmointi perustuu siis funktioihin ja aliohjelmiin, jotka suoritetaan tapahtumaohjatusti. Proseduurit sisältävät muuttujia, ehtoja ja silmukoita sekä mahdollisia upotettuja SQL –lauseita. Ohjelmakoodi muistuttaa hyvin paljon kirjoitettua englantia ja on siksi nopea oppia. Visual Basic antaa myös mahdollisuuden kirjoittaa sama toiminto usealla tavalla ja rivipohjaisena kielenä näitä vaihtoehtoja voi olla jopa kymmeniä. Visual Basicin työkaluvalikoima ja Accessin sovelluskehitin antavatkin nykyään hyvät mahdollisuudet tehokkaaseen tietojenkäsittelyyn.

4.2.1 Moduulit

Moduuli (Module) on Access tietokannan osa, joka sisältää VBA-ohjelmakoodia (Visual Basic for Applications). Moduuleita voi tietokannassa olla useita ja kukin niistä voi sisältää VBA-kielisiä funktioita (Function) ja aliohjelmia (Subprogram). (Sainio 2002, 12.)

Moduuleita on kahdenlaisia, yhteiskäyttö- ja luokkamoduuleita. Yhteiskäyttöiset moduulit tallennetaan tietokantaan omina objekteinaan ja ne näkyvät tietokantaikkunassa, kun objektilajiksi on valittu Moduulit (Modules). Luokkamoduulit liittyvät puolestaan tietokantaobjekteihin, esimerkiksi lomakkeisiin ja raportteihin, ja ne tallennetaan tietokantaobjektin yhteyteen. (Sainio 2002, 135.)

Tässä työssä on kaksi yhteiskäyttömoduulia, CADLINK- ja TABLELINK -moduuli sekä yksi luokkamoduuli Form_MENU -moduuli. CADLINK -moduuli(liite 2) sisältää proseduurit, joita tarvitaan scriptien tekemiseen, tiedon siirtämiseksi AutoCAD:iin. TABLELINK -moduuli sisältää jokaiselle tyyppikaavion tiedonsiirtotaululle oman kysely/päivitys –proseduurin, joka hakee tiedon lähdetietotauluista tiedonsiirtotauluun. Tämän proseduurin toiminnasta on kerrottu tarkemmin otsikon 4.3 alla. Form_MENU –moduuli sisältää käyttöliittymän eli lomakkeen toimintoihin liittyvät proseduurit. Käyttöliittymän toiminnasta ja käytännön toteutuksesta on kerrottu enemmän otsikon 2.5 alla.

4.3 SQL

IBM kehitti 1970-luvun puolivälissä relaatiomallin mukaista prototyyppitietokantaa projekti R –nimisessä hankkeessa. Tietokantakieleksi valittiin SEQUEL, jonka oli kehittänyt D. D. Chamberlainin tutkimusryhmä. Myöhemmin kieltä alettiin kutsua nimellä SQL (Strctured Query Language). (Hovi 2004, 17.)

Tässä työssä on käytetty ns. upotettua SQL-ohjelmointia. Upotettu SQL on SQL:n käyttöä toisesta ohjelmakoodista(Arola 1999 ,280). Tämä tarkoittaa käytännössä isäntäkieleen, joka tässä työssä on VBA, lisättyjä SQL-lauseita. Käyttämäni SQL on staattista, jonka Hovi (2004, 231) määrittelee seuraavasti. SQL-käskyjen rakenne on tiedossa ja koodattu ohjelmiin ja käskyt on myös optimoitu valmiiksi. Käskyn rakenne

on kiinteä (taulun nimet, sarakenimet, lajittelumääritykset jne.), ainoastaan isäntäkielen muuttujilla saadaan tiettyä joustavuutta. Staattinen SQL –ohjelmointi on tässä työssä järkevää. Tällä tavalla myös koodista tulee yksinkertaisempaa ja sitä kautta ohjelman laajentaminen ja muokkaaminen on helpompaa.

4.3.1 SQL:n ominaisuudet

Yksi SQL:n suurimmista eduista on, että se on todella monen alustan ja monen tuotteen kieli. Koska se on korkean tason tai neljännen polven kieli (4GL), sen avulla voi tehdä paljon asioita vain muutamalla rivillä ohjelmakoodia. (Arola 1999, 10.) SQL:llä ei kerrota, miten tietokantaoperaatiot tehdään, sillä kerrotaan vain, mitä halutaan tehtävän. SQL on siis ei-proseduraalinen kieli. Tyypillisesti SQL:n kysely tuo tulosjoukossa joukon tietueita (rivejä), kun taas proseduraaliset ohjelmat käsittelevät tietoja tietue kerrallaan. (Hovi 2004 ,16.)

SQL ei kuitenkaan ole pelkästään kyselyjen suorittamista ja tiedon hakua tietokannasta. Kuten Arola (1999, 11) toteaa. Sen avulla voit luoda tauluja, lisätä tietoja, poistaa tietoja, yhdistellä tietoja, käynnistää toimintoja tietokannan muutosten perusteella ja tallentaa kyselyjä ohjelmasi sisään tai tietokantaan.

Poistokyselyllä (Delete Query) voidaan taulukosta poistaa valitut tietueet. Poisto on peruuttamaton, poistetut tietueet saadaan takaisin vain syöttämällä tai lisäämällä ne uudestaan. Päivityskyselyllä (Update Query) voidaan muuttaa kerralla useita taulukon tietueita. Päivitettävät tietueet voidaan valita kuten muissakin kyselyissä. (Sainio 2002, 78, 80.)

SQL:n päivitys- ja poistokäskyt ovat voimakkaita ja näitä käytettäessä voi helposti tehdä virheitäkin (Hovi 2004, 128). Tämän vuoksi SQL –lauseita kirjottaessa täytyy olla huolellinen ja ennen niiden testaamista tulee varmistua, ettei mitään oleellisia tietoja voi kadota.

4.3.2 Kysely/päivitys -proseduuri

Kysely/päivitys -proseduuri tekee suurimman osan lähdetietojen käsittelystä. Näitä proseduureja on jokaiselle tiedonsiirtotaululle/tyyppikaaviolle omansa ja ne on

nimetty yhtäläisesti. Esimerkiksi tyyppikaavio T1 käyttää tiedonsiirtotaulua T1, jonka päivittämiseen käytetään proseduuria T1. Kysely/päivitys –proseduurit sijaitsevat TABLELINK –moduulissa, missä ei ole muuta ohjelmakoodia. Seuraavaksi käsitellään yhden tällaisen kysely/päivitys –proseduurin toimintaperiaatetta. Päivitykseen käytettävät proseduurit ovat kaikki saman mallin mukaisia, joten yleisen toimintaperiaatteen hahmottaminen on riittävää.

Ensimmäiseksi käyttäjän valitsemasta piiritunnus/kuvaus arvosta erotetaan pelkkä piiritunnus. Piiritunnus koostuu kahdesta numero-osasta ja keskellä olevasta kirjainosasta, joka poistetaan joitakin kyselyitä ajatellen. Piiritunnus ja kirjainosaton piiritunnus tallennetaan eri muuttujiksi, joiden perusteella kyselyitä aletaan toteuttaa.

Kyselyt suoritetaan MEASURING_POINT_LIST: lle, IO_LIST:lle, CABLE_LIST:lle ja INSTRUMENT_LIST:lle tai ARMATURE_LIST:lle. Viimeinen kysely määräytyy sen mukaan, onko kyseessä venttiili vai muu toimilaite. Kyselyiden tulokset tallennetaan eri muuttujiksi, jonka jälkeen kortti- ja virransyöttötyypin sekä kanavan mukaan tehdään vielä yksi kysely. Tällä kyselyllä määritetään väyläkortissa käytettävät liitinnumerot, jotka kyselyn tuloksista tallennetaan myös muuttujiksi. Nyt tyyppikaaviota varten tarvittavat tiedot ovat tulosmuuttujissa, joista ne on helppo siirtää tiedonsiirtotauluun.

Kyselyiden virheenkäsittely on toteutettu niin, että tulosmuuttujien arvoiksi asetetaan ”##Not found##”, mikäli kyselyn määrittelyillä ei löydy arvoja. Jos taas kyselyn määrittelyt antavat tuloksia, mutta joukossa on tyhjiä kenttiä eli Null-arvoja, asetetaan tulosmuuttujan arvoksi ”##Field is empty##”. Kuvajan (1995, 40) määritelmän mukaan Null-arvo tarkoittaa puuttuvaa tai tuntematonta arvoa. Se ei ole sama kuin nolla (0), tai tyhjä merkki (” ”).

Lopuksi tulosmuuttujat päivitetään tiedonsiirtotauluun päivityskyselyn avulla. Tässä jokaiselle tulosmuuttujalle määritetään päivitettävä taulu, sarake mihin tieto tulee ja rivi, jolta löytyy muuttujan tietoa vastaava kuvaus. Näiden määritysten perusteella saadaan kaikki kyselyistä saadut tiedot päivitettyä tiedonsiirtotauluun. Lopuksi informoidaan käyttäjää ”Linkkitaulu päivitetty.” –viesti-ikkunalla.

5 POHDINTA

Nykypäivän suunnittelutyö ei enää ole pelkkää kuvien piirtämistä. Piirtämisen ohessa käytetään monia apuohjelmia, joilla saadaan nopeutettua työn etenemistä. Valmiista apuohjelmistakaan ei aina löydy apua, vaan joudutaan tekemään paljon asioita käsin. Näissä tilanteissa suunnittelijan olisi hyvä hallita joitakin tietotekniikan perusteita käsin tehtävän työn helpottamiseksi.

Kehitystyöni tarkoituksena oli vähentää käsin tehtävää työtä tiettyjen asiakasprojektien osalta. Tässä onnistuttiin hyvin ja saatiin monta työvaihetta yhden painikkeen taakse. Näin suunnittelutyö on tehokkaampaa ja kirjoitusvirheet vähenevät, koska kertaalleen tarkastettu tieto siirretään suoraan kuviin. Ohjelmaa on tähän asti päästy testaamaan vain vanhoilla asiakasprojektin tiedoilla, mutta lähitulevaisuudessa on luvassa uusi projekti, jossa ohjelmaa päästään kokeilemaan konkreettisesti. Ohjelmakehityksen kannalta tämä on tärkeää, jotta tekemäni ohjelman toimivuudesta saadaan varmuus ja mahdolliset viat korjattua. Paljon työvaiheita jäi vielä automatisoimatta ja Insta Automation Oy:llä onkin ohjelman jatkokehityksestä selkeä visio, jota lähdetään tulevaisuudessa toteuttamaan. Tämä sisältää ainakin massakorvauksien lisäämistä ja muita päivitysominaisuuksia, jotka mahdollistavat entistä tehokkaamman suunnittelutyön projektin aikana.

Tätä työtä aloittaessani minulla ei ollut tietokannoista eikä Visual Basic –ohjelmoinnista minkäänlaista käsitystä. Nopeasti kuitenkin huomasin Microsoft Access tietokantaohjelman toimivan hyvin samankaltaisesti kuin muutkin Office –perheen tuotteet, joka antoi uskoa työn tekemiseen. Ohjelmaan tutustuttuani totesin Accessin tarjoamien työkalujen olevan minulle jotenkin vaikea selkosia, joten päätin hyödyntää niitä mahdollisimman vähän. Työn edistyessä totesin tämän olleen oikea ratkaisu. Accessin työkalut tarjoavat lähinnä yksinkertaisiin kyselyihin ja toimintoihin helppoja ja nopeita ratkaisuja, mutta koen niiden muokkaamisen kovin vaikeaksi verrattaessa Visual Basicilla kirjoitettuun koodiin.

Ohjelmoinnin opettelussa tärkeintä on koodin kirjoittamisen järjestelmällisyys. Visual Basicilla voi kirjoittaa saman asian monella eri tavalla, joten omanlaisensa kirjoitustyyli on hyvä säilyttää läpi koko koodin. Kun perustoiminnot ovat hallussa on helppo etsiä internetistä vaikeampiakin ohjelmanpätkiä, joita voi hyödyntää omassa

projektissaan. Kuitenkin on muistettava malttaa kirjoittaa koodi uudelleen sillä tavalla, jota itse olet käyttänyt, jotta koodi pysyy luettavana. Kun koodi on järjestelmällistä ja kommentoitu huolellisesti, pääsee ulkopuolinenkin siihen helposti sisälle, vaikka ei kyseistä koodia olisi aiemmin nähnytkään.

Tämän kehitystyön aikana minulle on kertynyt paljon uutta tietoa tietokannoista ja ohjelmoinnin perusteista. Uskon näistä tiedoista olevan hyötyä myös tulevaisuudessa ja ne antavat minulle mahdollisuuden entistä monipuolisempaan työskentelyyn tietoteknisempään suuntaan muuttuvassa yhteiskunnassa.

LÄHTEET

Arola, Jussi 1999. SQL-tietokantaohjelmointi. IT Press, Jyväskylä 1999, ISBN 951-826-042-7.

Hovi, Ari 2004. SQL-opas. Docendo Finland Oy, 1. painos, ISBN 941-846-228-3.

Kuvaja, Arto 1995. Visual Basic, tietokantaohjelmointi. Suomen ATK-kustannus Oy, Jyväskylä 1995, ISBN 951-762-290-2.

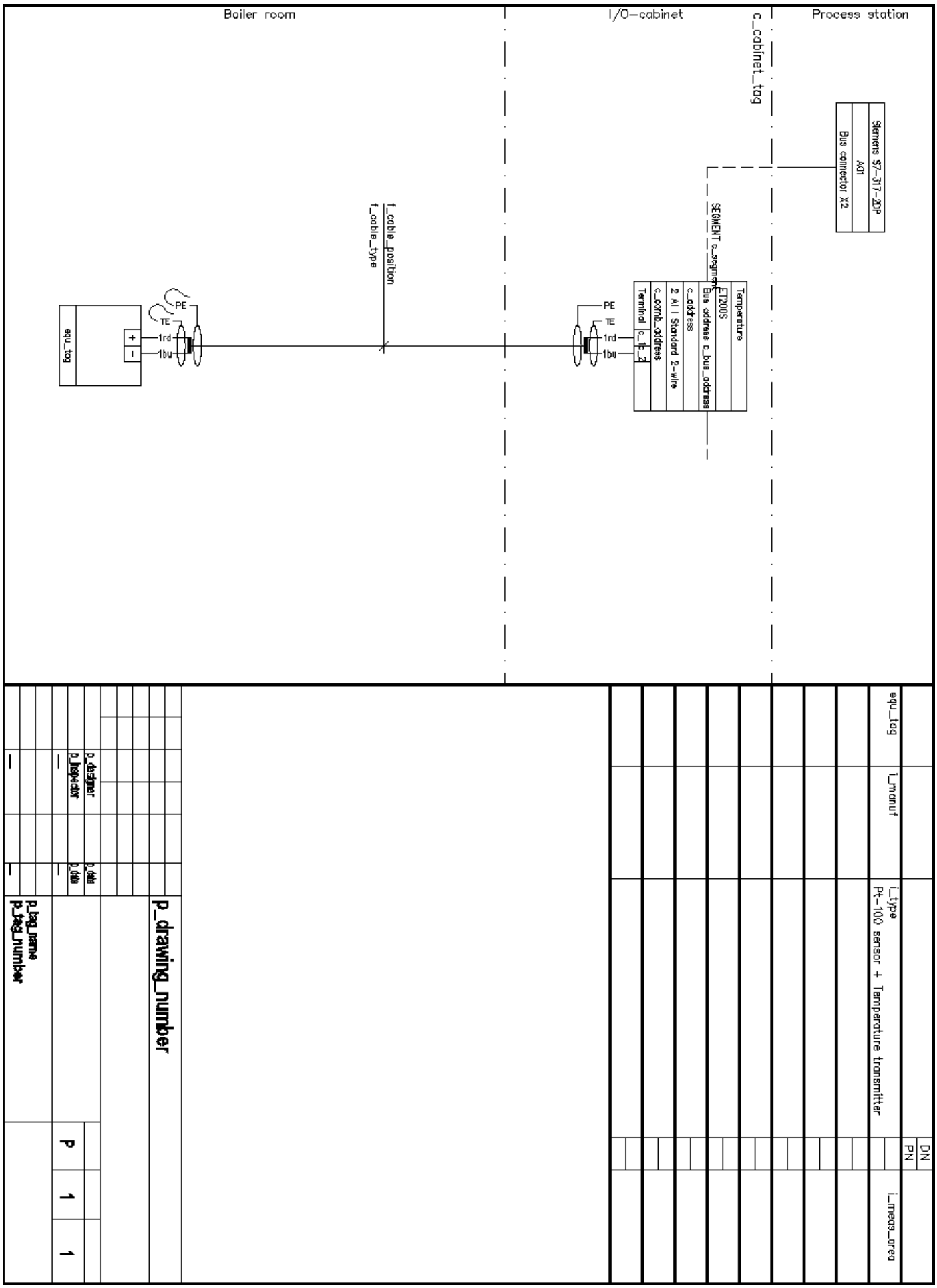
Laaksonen, Antti 2002. Visual Basic –opas. WWW-dokumentti.http://www.ohjelmointiputka.net/opask.php?tunnus=vbo_1 Ei päivitystietoa. Luettu 15.4.2011.

Sainio, Arto 2002. Access 2002. Docendo Finland Oy, ISBN 951-846-123-6.

Suominen, Erkki 1997. Office 97 –VBA-ohjelmointi. Suomen ATK-kustannus Oy, Jyväskylä 1999, 2. painos, ISBN 951-762-562-6.

LIITE 1.

Tyyppikaavio T1



```
Option Compare Database
```

```
'suorittaa "Suorita korvaus" -painikkeen toiminnan
Sub Drive()
    Dim linkPath As String

    'hakee LINK -kansion polun
    linkPath = GetLinkPath()
    'virheen käsittely "GetProjectPath" -funktiosta, mikäli polku on
    virheellinen
    If linkPath = "\LINK\" Then
        Exit Sub
    Else
        'tuhoaa mahdolliset vanhat .txt -tiedostot
        DeleteTxtFiles (linkPath)
        'tekee save.txt:n(sis. virh. käsittelyn) sekä myös tiedot.txt:n
        If MakeSaveTxt = False Then
            Exit Sub
        End If
        'tuhoaa mahdolliset vanhat .scr -tiedostot
        DeleteScrFiles (linkPath)
        'muuttaa .txt -tiedostojen päätteet . scr:ksi
        ChangeFilesToScr (linkPath)
        'avaa CAD:n, tallentaa kuvan "oikealla" nimellä ja suorittaa
        korvauksen
        OpenCadAndSave (linkPath)
    End If

End Sub
```

```
'hakee Projekti -polun ja määrittää sen perusteella LINK -kansiolle
polun
Function GetLinkPath() As String
    Dim link As Database
    Dim paths As DAO.recordSet
    Dim projectPath, projectLinkPath As String

    projectPath = GetProjectPath()
    projectLinkPath = projectPath & "\LINK\"

    GetLinkPath = projectLinkPath

End Function
```

```
'avaa PATHS -taulun ja määrittää siitä projekti -kansiolle polun
Function GetProjectPath() As String
    Dim link As Database
    Dim paths As DAO.recordSet
    Dim projectPath As String

    Set link = CurrentDb()
    Set paths = link.OpenRecordset("PATHS")
    'avaa virheikkunan mikäli polkua ei ole syötetty
```

```
    If IsNull(paths!Project_Path) Then
        MsgBox "Määritä Projekti -polku.", vbDirectory, "Ei polkua!"
        Exit Function
    Else
        projectPath = paths!Project_Path
        'tarkastaa Projekti -polun olemassaolon ja avaa virheikkunan
        mikäli polkua ei löydy
        If Not Dir(projectPath & "\LISTS.xls", vbDirectory) = _
        vbNullString Then

            GetProjectPath = projectPath
        Else
            MsgBox "VirheellinenProjekti -polku!",vbDirectory, _
            "Virheellinenpolku!"
            Exit Function
        End If
    End If

End Function
```

```
'avaa PATHS -taulun ja määrittää CAD -polun, avaa CAD:n ja ajaa scriptit
Sub OpenCadAndSave(linkPath As String)
    Dim link As Database
    Dim paths As DAO.recordSet
    Dim CADpath As String

    Set link = CurrentDb()
    Set paths = link.OpenRecordset("PATHS")
    'avaa virheikkunan mikäli polkua ei ole syötetty
    If IsNull(paths!AutoCAD_Path) Then
        MsgBox "Määritä CAD -polku.", vbDirectory, "Ei polkua!"
        Exit Sub
    Else
        CADpath = paths!AutoCAD_Path & "\"

        'tarkastaa CAD -polun olemassaolon ja avaa virheikkunan mikäli
        polkua ei löydy
        If Not Dir(CADpath& "acad.exe", vbDirectory) = vbNullString _
        Then

            ChDir (linkPath)
            'avaa CAD:n ja ajaa save.scr scriptin(save.scr avaa tie-
            dot.scr)
            X = Shell(CADpath & "acad.exe" & " /b " & "save.scr", 1)
        Else
            MsgBox "Virheellinen CAD -polku!",vbDirectory, _
            "Virheellinenpolku!"
            Exit Sub
        End If
    End If

End Sub
```

```
'tekee save.txt -tiedoston
Function MakeSaveTxt() As Boolean
    Dim link As Database
    Dim result As DAO.recordSet
    Dim fs, TextFile
    Dim TAGSelection, TAGNumber, savePathAndName, projectPath As String
    Dim typePath, savePath, linkTable,replacedLinkPath As String
    Dim linkPath, SQLQuery, saveAs As String

    'hakee menusta alasvetovalikon valinnan arvon
    TAGSelection = Form_MENU.SelectTAGNumber.Value

    'avaa virheikkunan mikäli kuvaa ei ole valittu
    If IsNull(TAGSelection) Then
        MsgBox "Valitse käsiteltävä kuva alasvetovalikosta.", _
        vbExclamation, "Eivalintaa!"
        Exit Function
    End If

    'erottaa menun valinnasta pelkän TAG Numberin
    TAGNumber = Left(TAGSelection, InStr(1, TAGSelection, "_") - 1)

    'hakee käytettävän linkkitaulun
    linkTable = GetLinkTable(TAGNumber)

    'avaa virheikkunan mikäli tyyppikuvaa(linkkitaulua) ei ole
    määritetty
    If linkTable = "Not found" Then
        MsgBox "Tyyppikuvaa ei ole määritetty, tarkasta _
        MEASURING_POINT_LIST -taulu.",vbExclamation, _
        "Puutteellinen määritys!"
        Exit Function
    End If

    Set link = CurrentDb()

    'hakee mittapiste -taulusta TAG Numberia vastaavan tallennusnimen
    SQLQuery = "SELECT Drawing_Number FROM MEASURING_POINT_LIST _
    WHERE TAG_Number = '" & TAGNumber & "';"

    Set result = link.OpenRecordset(SQLQuery)

    If IsNull(result("Drawing_Number")) Then
        saveAs = "Not found"
    Else
        saveAs = result("Drawing_Number")
    End If
    result.Close
    Set result = Nothing

    'avaa virheikkunan mikäli piirrustusnumeroa ei ole määritetty
    If saveAs = "Not found" Then
        MsgBox "Piirrustusnumeroa ei ole määritetty, tarkasta _
```

```
        MEASURING_POINT_LIST -taulu.",vbExclamation, _
        "Puutteellinen määritys!"
        Exit Function
    End If

    'hakeeprojekti -kansion
    projectPath = GetProjectPath()
    'määrittää tallennus -kansion
    savePath = projectPath & "\FILES\"

    ChDir (savePath)
    'poistaa mahdollisen vanhan kuvan ennen uuden tekemistä, jotta
    scripti ei pysähdy "korvataanko?" valintaan
    strFile = saveAs & ".dwg"
    Set fs = CreateObject("Scripting.FileSystemObject")
    On Error Resume Next
    fs.deletefile strFile

    'määrittää tyyppikuva -kansion ja kuvanimen
    typePath = projectPath & "\TYPEDWG\" & linkTable& ".dwg"
    'määrittää tallennus -kansion ja tallennusnimen
    savePathAndName = savePath& saveAs
    'hakee LINK -kansion polun
    linkPath = GetLinkPath()

    'korvaa Srxtext.vlx -ohjelmaa varten tarvittavasta polusta
    kenoviivat kahdeksi
    replacedLinkPath = Replace(linkPath, "\", "\\")

    ChDir (linkPath)
    'kirjottaa save.txt -tiedoston
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set TextFile = fs.createtextfile("save.txt", True)
        TextFile.WriteLine "open"
        TextFile.WriteLine """" & (typePath) & """"
        TextFile.WriteLine "saveAs"
        TextFile.WriteLine "2000"
        TextFile.WriteLine """" & (savePathAndName) & """"
        TextFile.WriteLine "(load """ & (replacedLinkPath) & _
        "Srxtext.vlx"")"
        TextFile.WriteLine "SCRIPT"
        TextFile.WriteLine """" & (linkPath) & "tiedot.scr"""
        TextFile.Close

    'siirtää kuvanimeä vastaavan linkkitaulu -tiedon tiedot.txt:n
    tekemistä varten
    MakeInformationTxt (linkTable)
    'todetaan funktion menneen "läpi" ilman virheitä(virheen korjaus
    "Drive" -funktiossa)
    MakeSaveTxt = True

End Function
```

```
'tekee tiedot.txt -tiedoston
Sub MakeInformationTxt(linkTable As String)
    Dim link As Database
    Dim recordSet As DAO.recordSet
    Dim fs, TextFile
    Dim linkPath As String

    Set link = CurrentDb()
    'linkTable -muuttuja tulee MakeSaveTxt -funktiosta
    Set recordSet = link.OpenRecordset(linkTable)

    'hakee LINK -kansiolle polun
    linkPath = GetLinkPath()

    ChDir (linkPath)

    'kirjottaa tiedot.txt -tiedoston
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set TextFile = fs.createtextfile("tiedot.txt", True)
        Do Until recordSet.EOF = True
            TextFile.WriteLine "SRXTEXT"
            TextFile.WriteLine "E"
            'vastaa linkkitaulussa korvattavaa tietoa, joka löytyy myös
            tyyppikuvasta
            TextFile.WriteLine (recordSet!Find)
            'vastaa kuvaan korvattavaa tietoa linkkitaulussa
            TextFile.WriteLine (recordSet!Replace_with)
            TextFile.WriteLine ""
            TextFile.WriteLine ""
            TextFile.WriteLine "D"

            recordSet.MoveNext
        Loop

        TextFile.WriteLine "ZOOM"
        TextFile.WriteLine "EXTENTS"
        TextFile.WriteLine "END"
        TextFile.Close

End Sub
```

```
'muuttaa .txt -tiedostojen päätteet . scr:ksi
Sub ChangeFilesToScr(linkPath As String)

    ChDir (linkPath)

    On Error Resume Next
    Name "save.txt" As "save.scr"
    On Error Resume Next
    Name "tiedot.txt" As "tiedot.scr"

End Sub
```

```
'tuhoaa mahdolliset vanhat .scr -tiedostot
Sub DeleteScrFiles(linkPath As String)

    ChDir (linkPath)

    strFile = "save.scr"
    Set fs = CreateObject("Scripting.FileSystemObject")
    On Error Resume Next
    fs.deletefilestrFile

    strFile = "tiedot.scr"
    Set fs = CreateObject("Scripting.FileSystemObject")
    On Error Resume Next
    fs.deletefilestrFile

End Sub
```

```
'tuhoaa mahdolliset vanhat .txt -tiedostot
Sub DeleteTxtFiles(linkPath As String)

    ChDir (linkPath)

    strFile = "save.txt"
    Set fs = CreateObject("Scripting.FileSystemObject")
    On Error Resume Next
    fs.deletefilestrFile

    strFile = "tiedot.txt"
    Set fs = CreateObject("Scripting.FileSystemObject")
    On Error Resume Next
    fs.deletefilestrFile

End Sub
```

```
'käytetään valittua kuvaa vastaavan linkkitaulun määrittämiseen
Function GetLinkTable(ByVal TAGNumber As String) As String
    Dim link As Database
    Dim result As DAO.recordSet
    Dim SQLQuery, linkTable As String

    Set link = CurrentDb()

    'hakee mittapiste -taulusta TAG Numberia(valintaa) vastaavan
    linkkitaulun nimen
    SQLQuery = "SELECT Loop_Typical FROM MEASURING_POINT_LIST WHERE _
    TAG_Number = '" & TAGNumber & "';"

    Set result = link.OpenRecordset(SQLQuery)

    'mikäli tyyppikuvaa(linkkitaulua) ei ole määritetty asetetaan
    linkTablen arvoksi "Not found", joka käsitellään myöhemmin virheenä
    If IsNull(result("Loop_Typical")) Then
        linkTable = "Not found"
```

```
    Else
        linkTable = result("Loop_Typical")
    End If
    result.Close
    Set result = Nothing

    GetLinkTable = linkTable

End Function
```